

**BUNDESREPUBLIK DEUTSCHLAND****PRIORITY  
DOCUMENT**SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH RULE 17.1(a) OR (b)

DE04/02462

**Prioritätsbescheinigung über die Einreichung  
einer Patentanmeldung****Aktenzeichen:**

103 52 172.0

REC'D 03 JAN 2005

**Anmeldetag:**

05. November 2003

WIPO

PCT

**Anmelder/Inhaber:**

Robert Bosch GmbH, 70442 Stuttgart/DE

**Bezeichnung:**

Verfahren und Vorrichtung zur Anpassung von Funktionen zur Steuerung von Betriebsabläufen

**IPC:**

G 06 F 9/445

**Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.**München, den 10. Dezember 2004  
**Deutsches Patent- und Markenamt****Der Präsident**

Im Auftrag

**Feust**

05.11.03 Sy

5

ROBERT BOSCH GMBH, 70442 Stuttgart

10

**Verfahren und Vorrichtung zur Anpassung von Funktionen zur Steuerung von Betriebsabläufen**

Stand der Technik

15

In der Funktionsentwicklung für Motronic-Steuergerätesoftware ist die Bypass-Anwendung ein Rapid-Prototypingverfahren um neue Steuergerätefunktionen zu entwickeln und zu testen. Als Entwicklungsverfahren kommen hierfür die beiden Anwendungen „Externer Steuergeräte-Bypass“ z.B. DE 101 06 504 A1 und „Interner Steuergeräte-Bypass“ z.B. DE 101 306 54 A1 zum Einsatz. Unabhängig beider Verfahren werden für die Anwendbarkeit, Eingriffe in der Steuergerätesoftware benötigt. Diese Eingriffe werden mit dem Begriff „Bypass-Freischnitte“ oder „Software-Freischnitte“ bezeichnet. Ein Bypass-Eingriff beschreibt genau die Stelle in einer Softwarefunktion, an der der Inhalt einer Steuergeräte-Variable nicht durch das Softwareprogramm, sondern über Umwege z.B. über eine Bypass-Softwarefunktion beschrieben wird. Bypass-Freischnitte sind sehr individuell und im Normalfall nicht Bestandteil eines Steuergeräte- Softwareprogrammes, da hierfür Speicher-Ressourcen verbraucht werden. Benötigt ein Funktionsentwickler ein Softwareprogramm mit Bypass-Freischnitten, so werden diese erst nach Beauftragung der Softwareentwicklung in die entsprechenden Softwarefunktionen eingebaut. Mit den geänderten Softwaremodulen und den Projektbestandteilen wird über einen Übersetzungs- und Linkvorgang ein neues Softwareprogramm erzeugt.

20

25

30

35

Nachteile des derzeitigen Verfahrens: Lange Durchlaufzeiten bis zur Verfügbarkeit des Softwareprogrammstandes. Hoher technischer und administrativer Aufwand zur Erstellung eines Softwareprogrammes mit Bypass-Freischnitten. Zusätzlicher Aufwand bei der Verwaltung der Module mit Bypass-Anteilen.

40

Nach aktuellem Wissensstand basiert ein vergleichbares Verfahren auf der Idee, die „Store“- Befehle (Schreibzugriff auf eine Steuergeräte-Variable), durch Sprungbefehl auf eine Unterfunktion zu ersetzen. Bei Mikrocontrollern mit gemischtem Befehlssatz (16-/32-Bit-CPU-Befehle) können u.A. die „Store“-Befehle 16-Bit-breit sein, da die

Adressierung indirekt über Adressregister erfolgt. Diese 16-Bit- breiten Befehle können für den Aufruf einer Unterfunktion nicht herangezogen werden, da der direkte, adressorientierte Aufruf einer Unterfunktion einen 32Bit- breiten Sprungbefehl erfordert. Somit ist das Verfahren nur bedingt einsetzbar und kann nur bei Mikroprozessoren mit „reinem“ 32-Bit-Befehlssatz angewendet werden.

#### Aufgabe der Erfindung

Ziel des neuen, erfindungsgemäßen Verfahrens ist es Bypass-Freischnitte ohne Source-Codeänderungen in ein vorhandenes Softwareprogramm einzubringen um damit die verschiedenen Rapid Prototyping-Verfahren schneller einsetzbar machen zu können. Ziel ist es Mikroprozessortypen mit einem Befehlssatz von 16-/32-Bit- breiten CPU-Befehlen mit diesem Verfahren zu unterstützen.

#### Beschreibung der Erfindung und Vorteile

Bei der Erfindung handelt es sich um ein „dynamisches Anhängen“ („Dynamic-Hooks“) von Software-Freischnitten ohne Source-Codeänderungen. Das hier beschriebene Verfahren ändert die Adressinformation von „Load“-Befehlen, ändert Funktionsaufrufe und fügt neue Programmcodes hinzu. Diese Änderungen werden an einem vorhandenen Softwareprogrammstand auf der Basis von gezielten HEX-Code-Modifikationen durchgeführt.

#### Hauptbestandteile des neuen Verfahrens „Dynamischer Software-Freischnitt“

- Verfahren zur Lokalisierung der zu ändernden Code-Sequenzen (Kapitel 2.4.1)
- Verfahren zur Code-Modifikation (Kapitel 2.4.2)
- Segmentierung der Speicherbereiche (Kapitel 2.4.3)
- Entwicklungsprozess zur Erstellung des Softwarecodes für die Eingriffe (Kapitel 2.4.4)
- Verfahren zum Einbinden des Software-Freischnittcodes

#### Vorteile des Verfahrens „Dynamischer Software-Freischnitt“

- Vorbereitung eines Softwarestandes für das Rapid Prototyping erfolgt ohne Mitwirkung der Softwareentwicklung
- geringer technischer und administrativer Aufwand, somit Reduzierung der Kosten

#### 2.4

Das nachfolgend dargestellte Verfahren basiert auf dem Einsatz von Mikrocontrollern, deren Befehlssatz 16-/32-Bit breite CPU-Befehle umfasst. Als exemplarisches Beispiel

dient hier z.B. der Mikrocontroller TriCore TC17xx (RISC/DSP/CPU) von Infineon, welcher Bestandteil eines Steuergeräts zur Steuerung von Betriebsabläufen, insbesondere bei einem Fahrzeug ist.

5 Das Verfahren kann aber auch bei „reinen“ 32-Bit-Mikroprozessoren (RISC-Prozessoren, z.B. PowerPC/MPC5xx) angewandt werden.

10 Grundsätzlich wird bei dem Verfahren davon ausgegangen, dass der Codegenerator des Compilers die Maschinen-Befehle linear anordnet, d.h., vor dem Zugriff auf eine globale Steuergeräte-Variable müssen die Adressinformationen der Variable in entsprechenden Adressregistern vorliegen. Dieser Sachverhalt ist bei den meisten Compilern gegeben.

#### 2.4.1 Verfahren zur Lokalisierung der zu ändernden Code-Sequenzen (Abbildung 1)

20 Ausgangspunkt hierfür ist ein Steuergeräte-Softwareprogramm, das im Format einer HEX-Datei zur Verfügung steht. Als weitere Datei dient eine Datenbeschreibungsdatei (ASAP), welche die Adressen der Steuergeräte-Variablen liefert. Mit einem Windows-Softwareprogramm wird die HEX-Datei disassembliert. Die entsprechenden Adressen der freizuschneidenden Steuergeräte-Variablen werden aus der Datenbeschreibungsdatei entnommen. Ein für das Verfahren erstelltes Windows-Softwareprogramm sucht ② im disassemblierten Programmcode ①, unter Zuhilfenahme der Adressinformation der Steuergeräte-Variable ⑦, die entsprechenden „Load-/Store“-Code-Sequenzen der Steuergeräte-Variable. Das Windows-Softwareprogramm ist ein  
25 Simulationsprogramm, welches die Registerinhalte nach jedem CPU-Befehl überprüft.

30 Wird ein „Store“-Befehl lokalisiert ③ und entspricht nach Ausführung des Befehls, der Inhalt des betroffenen Adressregisters, der Adresse, der zu ermittelnden Steuergeräte-Variable ④, dann liegt eine Fundstelle vor, an der die Steuergeräte-Variable beschrieben wird. Ausgehend von dieser Position, werden im disassemblierten Programmcode die Stellen zurückverfolgt, an denen dieses Adressregister mit der Adressinformation der Steuergeräte-Variable geladen wurde ⑤. Werden diese Stellen ermittelt ⑥, dann liegen weitere Fundstellen vor. Für das Verfahren sind diese gefundenen „Load“-Befehle entscheidend.

#### 2.4.2 Verfahren zur Code-Modifikation

##### **Modifikation der „Load“-Befehle (Abbildung 2)**

40 Beim neuen Verfahren werden die ermittelten „Load“-Befehle, bezogen auf den nachfolgenden „Store“-Befehl, durch Adressinformationen einer Pointer-Variable ④ ersetzt. Diese Pointer-Variable ③ wird über eine Entwicklungsumgebung erzeugt. Die Adresse der Pointer-Variable befindet sich in einem reservierten Freibereich des Speicher-Layouts für „Freischnitt-Variablen“ (RAM-Bereich). Die modifizierten „Load“-Befehle adressieren die gleichen Adressregister ⑤ wie es  
45 auch das Original-Softwareprogramm durchgeführt hat, der Unterschied besteht in der Adressinformation und in der Adressierungsart. Anmerkung: CPU-Befehle die

Adressinformationen in Adressregister laden sind 32-Bit breit. Der vorhandene 32-Bit-Befehl wird durch einen anderen 32-Bit-Befehl ersetzt.

### Modifikation der „Call“-Befehle (Abbildung 3)

Für jede lokalisierte „Load-/Store“-Fundstelle wird die Anfangsadresse der Funktion ermittelt, in der sich die Fundstellen befinden. Dies geschieht mit dem für das Verfahren entwickelte Windows-Softwareprogramm, welches ausgehend von der Position der „Load“-Fundstellen ②, den disassemblierten Programmcode zurückverfolgt, bis ein „RET“-Befehl gefunden wird ③. Mit der Adressinformation der Load-/Store-/Ret-Befehle ④ und den Informationen einer Adressliste, die aus einer ELF-Binärdatei oder MAP-Datei (Linker-Output) generiert wurde, werden die Anfangsadressen der betroffenen Funktionen ermittelt. Im gesamten Programmcode werden alle Aufruf-Befehle ⑤, die einen Sprung auf diese Funktion auslösen, lokalisiert ⑥ und durch Funktionsaufrufe der neuen Funktion ersetzt ⑦ (exemplarisch dargestellt mit „subfunction B“ und „subfunction C“).

### Freischnitt-Funktion (Hook-Funktion) (Abbildung 4)

Bevor ein Store-Befehl, auf eine, durch einen Pointer adressierte Steuergeräte-Variable sinnvoll beschreiben kann, muss die Pointer-Variable mit einer Variablen-Adresse adressiert werden ④. Für diese Initialisierung wird Programmcode benötigt. Hierfür wird für jede ermittelte Funktion, in der Fundstellen vorhanden sind, eine neue Freischnitt-Funktion (Hook-Funktion) erstellt ③. Diese Funktion befindet sich in einem reservierten Freibereich des Speicher-Layouts für den „Freischnitt-Code“ (Code-Bereich). Die Freischnitt-Funktion ist ein Ergebnis der Entwicklungsumgebung für „Dynamische Software-Freischnitte“. Durch die Änderung des Funktionsaufrufs wird anstatt der ursprünglichen Steuergeräte-Funktion nun die Freischnitt-Funktion aufgerufen ②. Der Funktionsaufruf der ursprünglichen Steuergerätefunktion liegt innerhalb der Freischnitt-Funktion ⑤. Weiterhin befindet sich in der Freischnitt-Funktion Programmcode zur Initialisierung der Pointer-Variablen ⑥. Soll das Schreiben auf eine Steuergeräte-Variable unterbunden werden, so muss die Pointer-Variable, mit der Adresse einer Ersatz-Variable initialisiert werden ⑦. Der Store-Befehl der geänderten Softwarefunktion schreibt nun auf diese Ersatz-Variable. Die Originale Steuergeräte-Variable kann dann mit Werten einer Bypass-Funktion beschrieben werden.

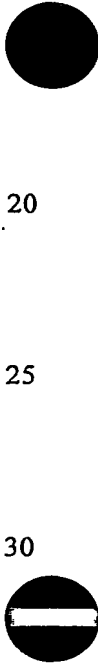
### Segmentierung der Speicherbereiche (Abbildung 5)

Für das Freischnittverfahren werden eigene Speicherbereiche im Speicher-Layout des Steuergeräte-Softwareprogrammes benötigt. Ähnlich wie beim „internen Steuergeräte-Bypass“ beansprucht das Verfahren Freibereiche für Code, Daten und RAM.

Im Code-Bereich liegen die Freischnitt-Funktionen ②, im Datenbereich sind Grössen definiert, über die ein Umschaltmechanismus zur Initialisierung der Pointer realisiert ist ③. Im RAM-Bereich sind Pointervariablen und administrative Grössen definiert ④.

### 2.4.4 Entwicklungsprozess „Dynamischer Software-Freischnitt“ (Abbildung 6)

Die Erzeugung des Software-Freischnittcodes erfolgt automatisch über eine für das Verfahren erstellte Entwicklungsumgebung.

- 
- ① HEX-Code-Datei. Beinhaltet Maschinencode im Intel-Hex- oder Motorola-S19-Format
  - ② Beschreibungsdatei der Applikationsdaten. Beinhaltet z.B. Adressen und Umrechnungsformeln von RAM-Variablen, Kennlinien, Kennfelder
  - ③ Disassembler-Programm zur Konvertierung des Maschinencodes in lesbare Assembler-Befehle
  - ④ Disassemblierter Programmcode. Dient als Input für den Registersimulator
  - ⑤ Anwender gibt die RAM-Variablen vor zu denen die Load-/Store-Befehle gesucht werden sollen
  - ⑥ Register-Simulator. Liest sequentiell alle Opcodes und überprüft die Registerinhalte.
  - ⑦ ELF-Binär-Datei (Linker-Outputdatei) mit Adressen von Funktionen und Variablen
  - ⑧ Referenzdatenbank. Dient zur Auflösung offener Referenzen
  - ⑨ Dynamic-Hooks-Entwicklungsumgebung. Steuert alle Vorgänge zur Erzeugung des Freischnitt-Codes
  - ⑩ Automatisch generierter Source-Code der Freischnitt-Funktionen und Patch-Befehle zur Modifikation der Load-Befehle in der Basissoftware
  - ⑪ Compiler-/Linker und Werkzeuge zur Generierung der Daten-Beschreibungsdatei
  - ⑫ Arbeitsprodukt der Dyn.Hooks-Entwicklungsumgebung. Hex-Code beinhaltet die übersetzten Freischnitt-Funktionen sowie den Code zur Änderung der Loadbefehle
  - ⑬ Arbeitsprodukt der Dyn.Hooks-Entwicklungsumgebung. A2L-Datei beinhaltet die neuen Applikationsdaten mit Adressen zur Steuerung des Freischnittcodes
  - ⑭ Hex-/A2L-Merger. Vorgang, bei dem die Arbeitsprodukt der Dyn.Hooks-Entwicklungsumgebung mit der originalen Projektsoftware verbunden werden
  - ⑮ Arbeitsprodukt der Dyn.Hooks-Entwicklungsumgebung. Hex-Code beinhaltet die Original-Software mit den Patches und den Freischnittcode
  - ⑯ Arbeitsprodukt der Dyn.Hooks-Entwicklungsumgebung. A2L-Datei beinhaltet alle Applikationsdaten der Original-A2L-Datei und die Freischnitt-Verstellgrössen

#### **2.4.5 Verfahren zum Einbinden des Software-Freischchnittcodes**

Das Einbinden des Software-Freischchnittcode wird durch einen HEX-Merge durchgeführt. Bei dieser Aktion werden die Ergebnisse der Entwicklungsumgebung für das „Dynamische Freischnittverfahren“ (HEX-Code), in die Freibereiche des originalen Softwareprogrammes (HEX-Code) kopiert. Das Verfahren ist ähnlich aufgebaut, wie das des „Internen Steuergeräte-Bypass“.

05.11.03 Sy

5

ROBERT BOSCH GMBH, 70442 Stuttgart

10

Ansprüche

15

1. Verfahren zur Anpassung von Funktionen zur Steuerung von Betriebsabläufen, wobei die Funktionen auf wenigstens eine globale Variable wenigstens eines Programms zur Steuerung zurückgreift und dieser globalen Variable eine Adressinformation zugeordnet ist, welche in wenigstens einem Speichermittel vorliegt, wobei diese Adressinformation der globalen Variablen durch Ladebefehle aus dem Speichermittel geladen wird dadurch gekennzeichnet, dass die Adressinformation der globalen Variable des Ladebefehls ersetzt wird.

20

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass die Adressinformation der globalen Variable durch die Adressinformation einer Zeigervariable ersetzt wird

3. Verfahren nach Anspruch 2, dadurch gekennzeichnet, dass die Adressinformation der Zeigervariable in einem reservierten Speicherbereich vorliegt.

25

4. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass aus der Adressinformation eine Anfangsadresse der Funktion ermittelt wird.

30

5. Verfahren nach Anspruch 1 oder 4, dadurch gekennzeichnet, dass die Funktionen zur Steuerung von Betriebsabläufen durch Ersetzen der Adressinformation durch Zusatzfunktionen ersetzt werden.

6. Vorrichtung zur Durchführung eines Verfahren nach einem der Ansprüche 1 bis 5.

35

7. Steuergerät zur Steuerung von Betriebsabläufen mit einer Vorrichtung zur Durchführung des Verfahrens nach einem der Ansprüche 1 bis 5.

8. Computerprogramm zur Ausführung des Verfahrens nach einem der Ansprüche 1 bis 5.

40

9. Computerprogrammprodukt mit Programmcodemitteln zur Ausführung des Verfahrens nach einem der Ansprüche 1 bis 5.



05.11.03 Sy

5

ROBERT BOSCH GMBH, 70442 Stuttgart

10

**Verfahren und Vorrichtung zur Anpassung von Funktionen zur Steuerung von Betriebsabläufen**

Zusammenfassung

15

Verfahren und Vorrichtung zur Anpassung von Funktionen zur Steuerung von Betriebsabläufen, wobei die Funktionen auf wenigstens eine globale Variable wenigstens eines Programms zur Steuerung zurückgreift und dieser globalen Variable eine Adressinformation zugeordnet ist, welche in wenigstens einem Speichermittel vorliegt, wobei diese Adressinformation der globalen Variablen durch Ladebefehle aus dem Speichermittel geladen wird und die Adressinformation der globalen Variable des Ladebefehls ersetzt wird.

20

(Figur 6)



Abbildung 1:

Verfahren zur Lokalisierung der „Load-/Store“-Befehle

- 1 disassemblierter Programmcode
- 2 Vorgang zur Lokalisierung der "Store"-Befehle
- 3 "Store"-Befehl gefunden
- 4 Adressinhalt prüfen
- 5 Vorgang zur Rückverfolgung der Registerinhalte, Lokalisierung der "Load"-Befehle
- 6 "Load"-Befehle gefunden
- 7 Adresse der Steuergeräte-Variable

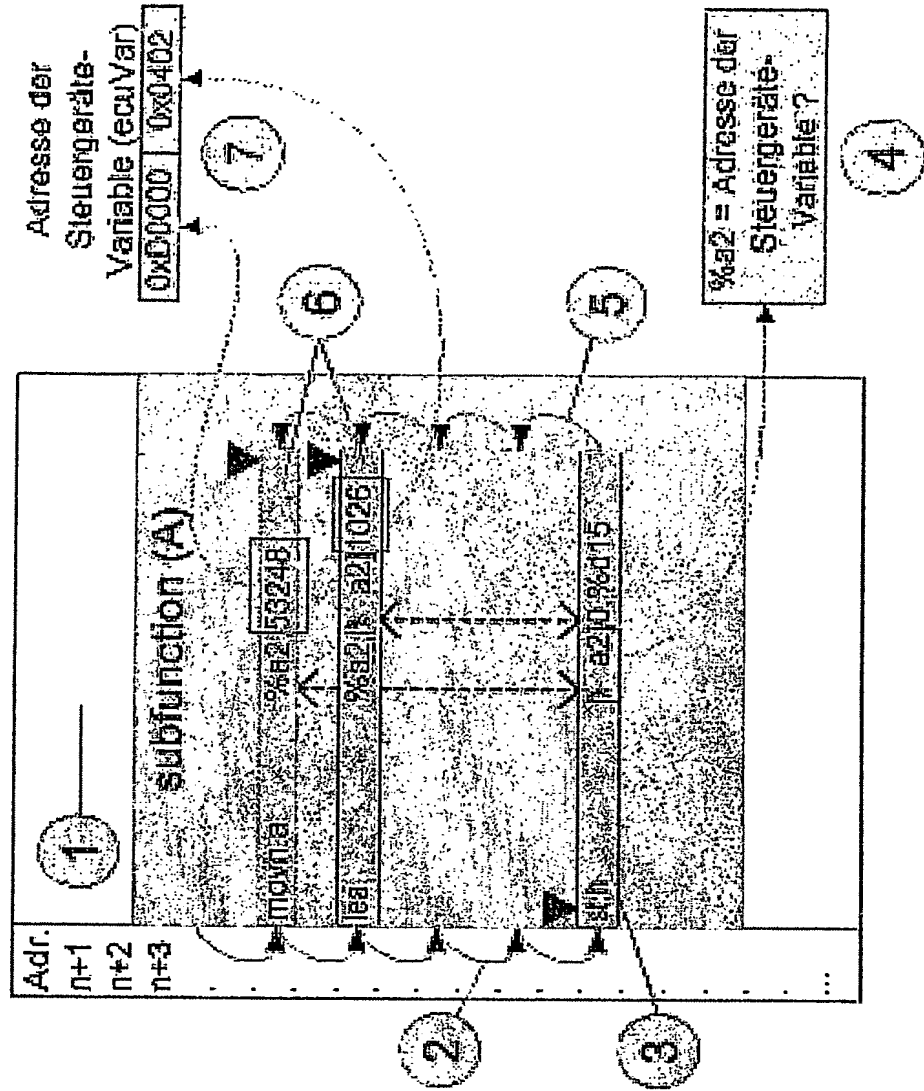
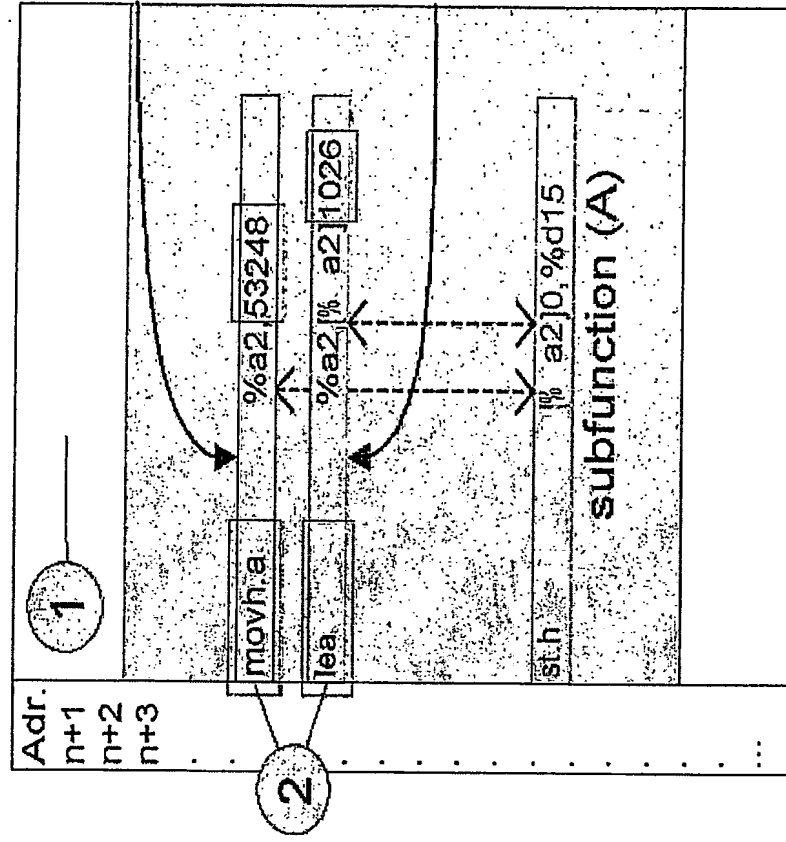


Abbildung 2:

Verfahren zur Modifizierung der „Load“-Befehle

- 1 Softwareprogramm (org. HEX-Code)
- 2 Fundstelle der "Load"-Befehle
- 3 Pointer-Variable
- 4 Modifizierte "Load"-Befehle mit den Adressinformationen der Pointer-Variable
- 5 Adressregister über die adressiert wird



### Abbildung 3: Verfahren zur Modifizierung der „Call“-Befehle

- 1 disassemblierter HEX-Code
- 2 Vorgang zur Rückverfolgung bis zum "RET"-Befehl
- 3 "RET"-Befehl der vorherigen Softwarefunktion
- 4 Fundstelle, erster Befehl nach dem "RET", entspricht der Anfangsadresse der Funktion
- 5 Unterfunktionen, die den Aufruf der Funktion "subfunction A" beinhalten
- 6 Vorgang zur Lokalisierung der Funktionsaufrufe
- 7 Aufruf-Befehle der "Subfunktion A". Call-Befehl gefunden
- 8 Modifizierte "Call"-Befehle für den Aufruf der Freischnitt-Funktion
- 9 Freischnitt-Funktion

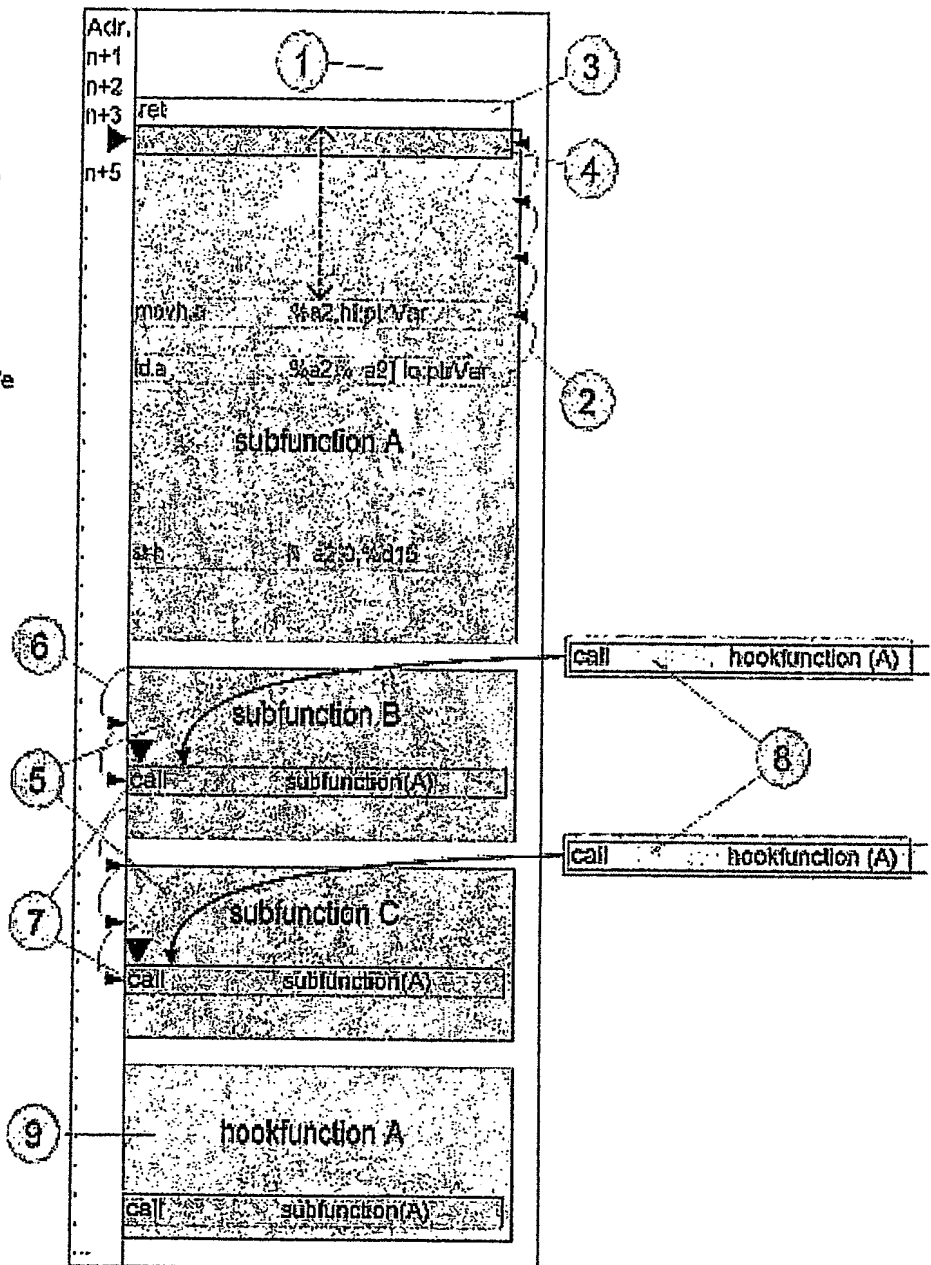


Abbildung 4:

# Aufruf und Funktionsweise der Freischnitt-Funktion

- 1 Softwareprogramm (org. HEX-Code)
- 2 Modifizierter Aufruf der Freischnitt-Funktion
- 3 Freischnitt-Funktion
- 4 Initialisierung der Pointer-Variable
- 5 Aufruf von "subfunction A"
- 6 Individuelle Initialisierung der Pointer-Variable z.B. für den Bypass-Betrieb
- 7 Initialisierung mit der Adresse einer temporären Variable
- 8 Unterfunktion "subfunction A"
- 9 "Store"-Befehl auf die Variable, die über \*ptrVar adressiert ist

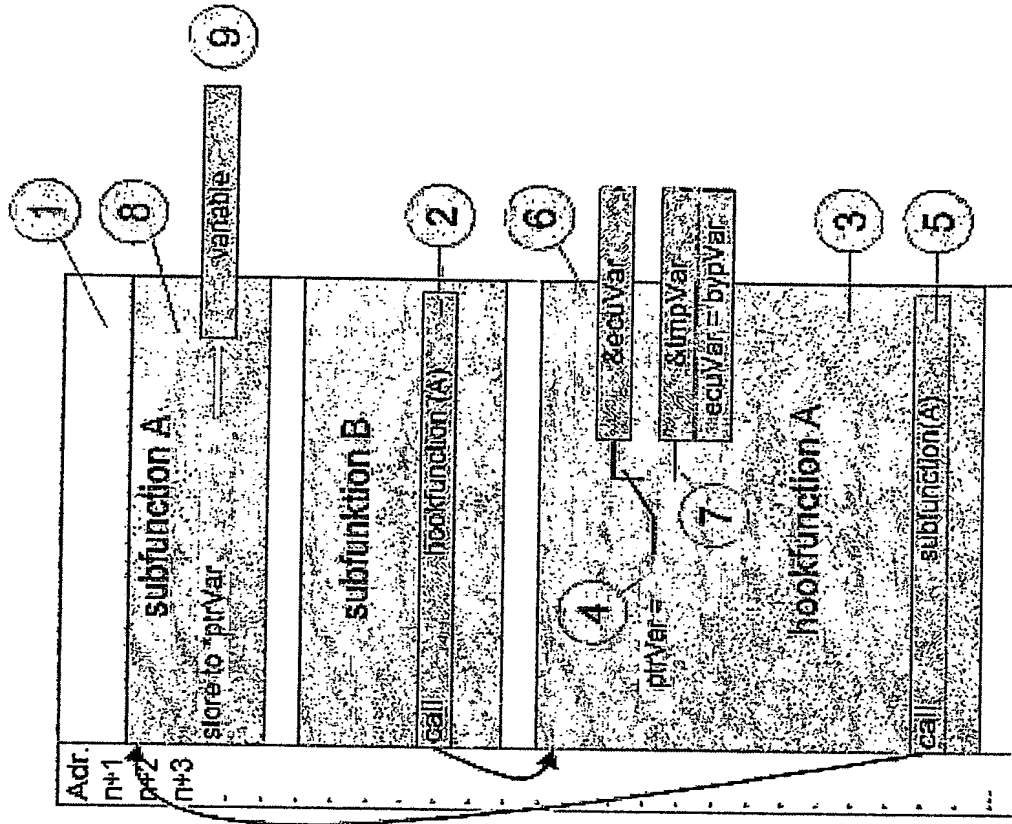
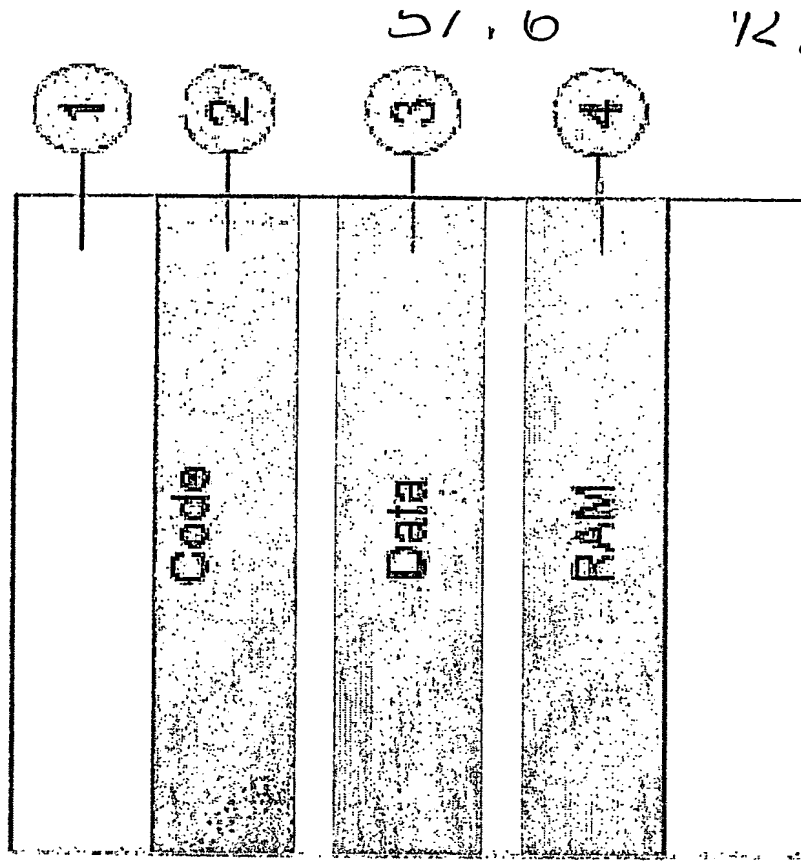




Abbildung 5:

Speicher-Segmentierung für das „Dynamische Freischnitt-Verfahren“



1 Speicher-Layout des Steuergeräte-  
Softwareprogrammes

2 Code-Bereich für Freischnitt-Code

3 Datenbereich für administrative Freischnitt-  
daten (Aktivierungsschalter)

4 RAM-Bereich für administrative Freischnitt-  
RAM-Größen, Pointer-Variablen

**Abbildung 6:**  
**Entwicklungsprozess „Dynamischer Software-Freischnitt“**

